DTD based costs for Tree-Edit distance in Structured Information Retrieval

Cyril Laitang, Karen Pinel-Sauvagnat, Mohand Boughanem

DTD Based Costs for Tree-Edit Distance in Structured Information Retrieval

Cyril Laitang, Karen Pinel-Sauvagnat, and Mohand Boughanem

IRIT-SIG,
118 route de Narbonne,
31062 Toulouse Cedex 9, France
{laitang,sauvagnat,boughanem}@irit.fr

Abstract. In this paper we present a Structured Information Retrieval (SIR) model based on graph matching. Our approach combines content propagation, which handles sibling relationships, with a document-query structure matching process. The latter is based on Tree-Edit Distance (TED) which is the minimum set of insert, delete, and replace operations to turn one tree to another. To our knowledge this algorithm has never been used in ad-hoc SIR. As the effectiveness of TED relies both on the input tree and the edit costs, we first present a focused subtree extraction technique which selects the most representative elements of the document w.r.t the query. We then describe our TED costs setting based on the Document Type Definition (DTD). Finally we discuss our results according to the type of the collection (data-oriented or text-oriented). Experiments are conducted on two INEX test sets: the 2010 Datacentric collection and the 2005 Ad-hoc one.

1 Introduction

Structured information retrieval (SIR) aims at ranking document parts instead of whole documents. For this purpose, SIR exploits document structure to focus on the user needs and to return XML elements that are both exhaustive and specific to his/her need. Structured document collections are of two types: some are strongly structured and contain textual information that can be seen as database records (they are called *data-oriented*) and others are more loosely structured but contain content designed to be read by humans (they are called *text-oriented*). Whatever the considered type of collection, queries on these collections can be expressed using both content (keywords) and structural constraints about content location. These queries are called *content and structure* (CAS) queries. Both XML documents and CAS queries can be naturally represented through trees where nodes are elements and edges hierarchical dependencies. Text content is located in the leaves and element names are the nodes labels. An example of an XML document and a CAS query is given in figure 1.

In the literature two types of approaches were proposed to handle document structure regardless of content. The first one is relaxation [1] [4] [7]. In these approaches, the main structure is atomized into a set of weighted node-node relationships. These weights are the distance between nodes in the original

structure. The second family is related to subtree extraction. The aim is to extract a particular subtree representative from the overall document structure. One of these approaches is the *lowest common ancestor* (LCA) in which the tree is rooted by the first common ancestor of two or more selected nodes [5]. In SIR this approach aims at scoring structure by finding subtrees where all the leaves contain at least one term of the query [3].

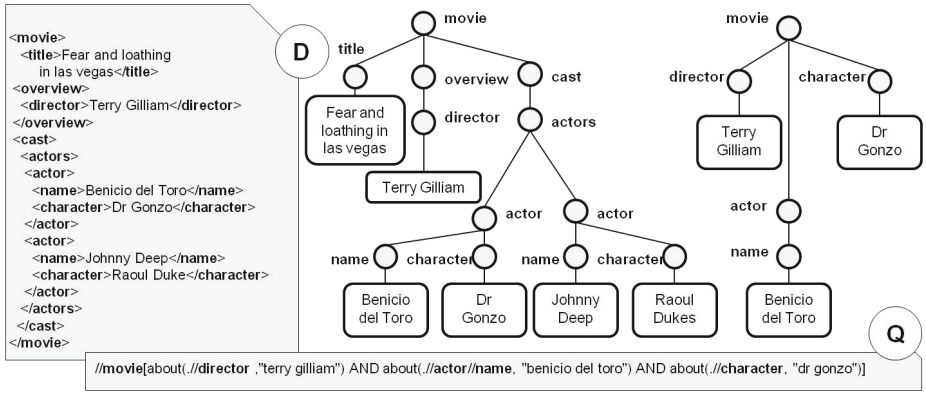

Fig. 1. Tree representation of an XML document and a query in which *we want a "movie" directed by "Terry Gilliam" with the actor "Benicio del Toro" and with a character named "Dr Gonzo".*

To the best of our knowledge, only a few approaches use the graph theory and none uses Tree-Edit Distance (TED) to evaluate similarity between the documents and query trees. One can however find approaches using string edit distance on path instead of trees [20]. The model we propose in this paper evaluates the relevance of an element with respect to a query using both content scoring propagation and a structure similarity measure based on TED. The research questions we address are the following:

- is TED useful for Structured Information Retrieval?
- as the effectiveness of TED mainly depends on the removing and relabeling costs and as these costs are often fixed in graph theory, how effective are edit costs if they are computed according to some document features?
- does the collection type (data or text-oriented) on which TED is applied affect the results?

The rest of this paper is organized as follows: section 2 gives a brief introduction on TED algorithms and details our structured oriented model. In section 3, in order to evaluate our TED based SIR model, we conducted some experiments on two different test sets. Finally, the impact of the collection type on the SIR process is discussed in section 4.

2 Tree-Edit Distance for Structural Document-Query Matching

We assume that a query is composed of content (keywords) and structure conditions, as shown in figure 1. The document-query relevance is evaluated by considering content and structure separately before combining them to rank relevant elements. In this section, we first describe the query-document content evaluation and we then detail our structure matching algorithm based on TED.

2.1 Content Relevance Score Evaluation

Our content score evaluation of a node n in a document is a three steps process. First, for each keyword in the query, we use a $tf \times idf$ (Term Frequency $\times$ Inverse Document Frequency [11]) formula to score the document leaf nodes according to query terms contained in content conditions. Then the scores are propagated through the document tree structure. Regarding the propagation, our intuition is that the score of an inner node n must depend on three operands. First, it must contain its intermediate score $p(n)$ computed as the mean of its leaves score. Then it must take into account its neighbors, particularly its siblings and parent nodes, as they could be seen as contextualization parameters. Indeed, relevant elements are more likely to be found in an overall relevant document [25]. Based on these constraints we define the content score $c(n)$ of a node n as the *intermediate content score of the element itself plus its siblings intermediate score plus its parent's score*. Recursively, and starting from the document root, $c(n)$ is computed as follows:

$$c(n) = \begin{cases} \underbrace{p(n)}_{(i)} + \underbrace{\frac{\sum_{b \in siblings(n)} p(b)}{\mid siblings(n) \mid}}_{(ii)} + \underbrace{\frac{c(a_1) - p(n)}{\mid children(a_1) \mid}}_{(iii)} & \text{if } n \neq root \\ \overbrace{p(n)} & \text{otherwise} \end{cases} \quad (1)$$

(i) is the *intermediate content score* $p(n) = \frac{\sum_{x \in leaves(n)} p(x)}{|leaves(n)|}$ evaluated using a $tf \times idf$ formula; (ii) is the *siblings score* computed as the mean of the node n siblings intermediate content scores $p(b)$; (iii) is the *parent score*, evaluated with $c(a_1)$ the final score of the father a_1 divided by the number all of its children nodes.

2.2 Structure-Based Relevance Score Evaluation

The second part of our approach is the structure score evaluation. This process follows three steps. The first one is subtrees selection and extraction. The second step is the structural evaluation through TED. The final step is then the structure score normalization. As our structure similarity evaluation process relies on Tree-Edit Distance we will first overview some state-of-the-art algorithms.

Tree-Edit Distance Algorithms. Two graphs are called isomorphic if they share the same nodes and edges. Evaluating their isomorphism is called graph matching. We make the distinction between approximate and exact matching. While the first one attempts to find a degree of similarity between two structures, exact matching validates the similarity. Due to the context of our work (Information Retrieval), we focus on approximate matching. There are three main families of approximate tree matching: *edit distance*, *alignment* and *inclusion*. We used Tree-Edit Distance (TED) as it has the most generalized application field of the three main families [6]. TED algorithms [23] generalize Levenshtein *edit distance*[16] to trees. The similarity is the minimal set of operations (adding, removing and relabeling) to turn one tree to another. Later, Klein et al. [14] reduced the overall complexity in time and space by splitting the tree structure based on the heavy path (defined in the following paragraph). Finally Touzet et al. [8] used a decomposition strategy to dynamically select the best nodes to recurse on between rightmost and leftmost, which reduces the number of subtrees in memory. Regarding the costs, a common practice is to use *apriori* fixed costs for the primitive operations [17] [18], i.e: 1 for removing a node, 0 for relabeling a node by another if their tags are similar and 1 otherwise. However as these costs strongly impact the isomorphism evaluation one may find non-deterministic approaches that try to estimate these costs using training techniques [19] [18].

To our knowledge, TED has never been used in SIR, and this is what we propose in this paper.

Subtree Extraction. Our approach uses minimal subtrees representing all the relevant nodes (having $p(n) > 0$) labeled with a label contained in the query as input for the matching process. This representation is considered as minimal as it prunes all irrelevant branches regarding to the query while keeping original internal document relationships in the considered branches.

These subtrees S are created from the combination of all the paths from the deepest relevant nodes to the highest node in the hierarchy, all containing a label from the query. These paths are then merged and rooted by a node having the same label l than the query root ($root(Q)$). Formally a subtree S is composed of all nodes n from the document tree D having descendants $des(n)$ and ancestors $anc(n)$ sharing a label with the query ($l(n) \in \{l(Q)\}$):

$$S = \{n \in D, P(n) > 0 \wedge l(root(Q)) \in \{l(anc(n))\} \wedge \exists d \in des(n)/l(d) \in \{l(Q)\}\} \quad (2)$$

The different steps of subtrees extraction are illustrated in figure 2. Starting from leaves with a score > 0 (step 1), branches are extracted from the first ancestor matching a label of the query (in our case "director", "name" and "character") to the query root label (step 2). These paths are merged into one subtree (step 3). These steps reduce the average subtrees size and increase our model efficiency as TED runtime strongly depends on the input trees cardinality.

Edit Distance Optimal Path. TED is a way of measuring similarity based on the minimal cost of operations to transform one tree to another. The number

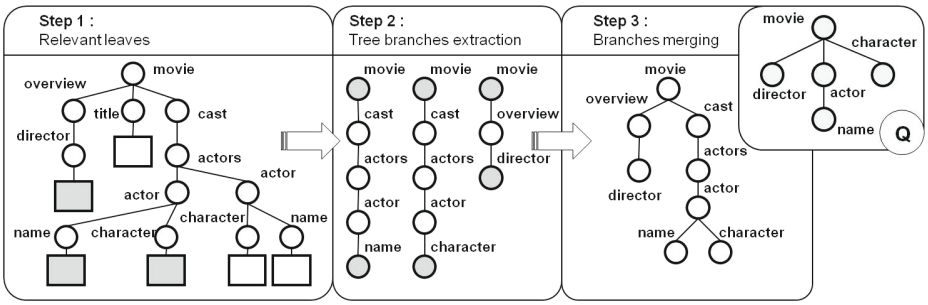

Fig. 2. Illustration of the different steps of our minimal subtree extraction

of subtrees stored in memory depends on the direction we choose when applying the operations. Our algorithm is an extension of the optimal cover strategy from Touzet et al. [8]. The difference is that the optimal path is computed with the help of the *heavy path* introduced by Klein et al. [14]. The heavy path is the path from root to leaf which passes through the rooted subtrees with the maximal cardinality[1]. This means that selecting always the most distant node from this path allows to create the minimal set of subtrees in memory during the recursion: this is the *optimal cover strategy*. Formally a heavy path is defined as a set of nodes $(n_1, ..., n_z)$, with $T(x)$ the rooted tree in x satisfying:

$$\forall (n_i, n_{i+1}) \in heavy \begin{cases} n_{i+1} \in children(n_i) \\ \forall x \in children(n_i), x \notin \{n_{i+1}\}, \mid T(n_{i+1}) \mid \geq \mid T(x) \mid \end{cases} \tag{3}$$

This strategy is used on the document and query in our TED algorithm :

Algorithm 1: Edit distance using optimal paths

```
d(F, G, p_F, p_G) begin
    if F = ∅ then
        if G = ∅ then
            return 0;
        else
            return d(∅, G - O_G.get(p_G)), p_F, inc(p_G)) + c_del (O_G.get(p_G));
        end
    end
    if G = ∅ then
        return d(F - O_F.get(p_F)), ∅, inc(p_F), p_G) + c_del (O_F.get(p_F));
    end
    a = d(F - O_F.get(p_F), G, inc(p_F), p_G) + c_del (O_F.get(p_F));
    b = d(F, G - O_F.get(p_F), p_F, inc(p_G) + c_del (O_G.get(p_G));
    c = d(T(O_F.get(p_F)) - O_F.get(p_F), T(O_G.get(p_G)) - O_G.get(p_G), inc(p_F), inc(p_G)) +
    d(F - T(O_F.get(p_F)), G - T(O_G.get(p_G)), next(p_F), next(p_G)) + c_match (O_F.get(p_F),
    O_G.get(p_G));
    return min(a, b, c);
end
```

F, G are two forests (i.e. the document and the query as first input), p_F and p_G are positions in O_F and O_G the *optimal paths* (i.e. paths of the *optimal*

[1] A tree cardinality is its number of nodes.

cover strategy). Function $O.get(p)$ returns the node in path O corresponding to position p and function $inc()$ increments this position.

Edit Distance Costs Evaluation. As seen in the beginning of this section, TED operation costs are generally set to 1 for removing, to 0 for relabeling similar tags and to 1 otherwise [23] which is sufficient for evaluating relatively similar trees. However in our approach document trees are usually larger than query trees which means that the removing cost should be assigned lower values. In addition TED costs should be adapted to the considered collection: some nodes may have higher associated costs than others depending on their informativeness in the collection. There are two constraints in estimating these costs. First, as relabeling is equivalent to removing and then adding a node, its cost should be at less or equal to two removing operations. Second, an IR model should be efficient as well as effective. For this reason we need to get the estimation of these costs for a minimal computation time. For all these reasons we propose to use the DTD (Document Type Definition) of the documents which contains all the transition rules between the document elements. We use this DTD to create an undirected graph representing all the possible transitions between elements (figure 3). We choose it to be undirected in order to make elements strongly connected. The idea is that the less degree a node has the less its removing cost should be.

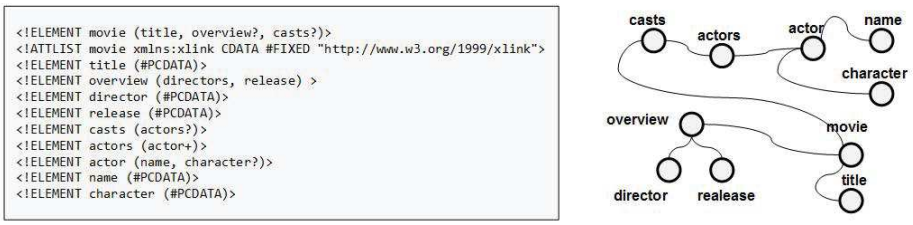

Fig. 3. Example of a "movie" DTD with its corresponding graph

As some collections can come up with several DTDs we create one graph for each of them and one final merged on shared labels if they exist (this explains our choice to represent the DTD graph in figure 3 as a graph and not as a tree). This merged graph is then used when the query is not explicit enough and is conform to more than one DTD.

In order to process the relabeling cost $c_{match}(n_1, n_2)$ of a node n_1 by a node n_2, respectively associated with tags t_1 and t_2, we seek the shortest path $sp()$ in these DTD graphs through a Floyd-Warshall [9] algorithm. This allows to overcome the cycle issues. We divide this distance by the longest of all the shortest paths that can be computed from this node label to any of the other tags in the DTD graph. Formally :

$$c_{match}(n_1, n_2) = \frac{sp(t_1, t_2)}{max(sp(t_1, t_x))} \forall x \in DTD \tag{4}$$

Similarly the removing cost is the highest cost obtained from all the relabeling costs between the current document node and all of the query nodes. Formally

$$c_{del}(n_1) = max(\frac{sp(t_1, t_y)}{max(sp(t_1, t_x))}) \forall x \in DTD; \forall y \in Q \tag{5}$$

The final structure score $s(n)$ of a node n is evaluated according to the TED $d(S, Q)$ of subtree S (S is the subtree rooted in n) and query Q divided by S cardinality. This normalization is done in order to reduce the influence of subtree size on the final score.

$$s(n) = 1 - \frac{d(S, Q)}{\mid S \mid} \tag{6}$$

2.3 Final Structure and Content Combination

The final score $score(n)$ for each candidate node n is evaluated through the linear combination of the previously normalized scores $\in [0, 1]$. Formally, with $\lambda \in [0, 1]$:

$$score(n) = \lambda \times c(n) + (1 - \lambda) \times s(n). \tag{7}$$

3 Experiments and Results

The experiments we conducted are based on two collections of the *Initiative for the Evaluation of XML Retrieval* (INEX) campaign which is the reference evaluation campaign for SIR models. We choose these test sets as they both contain strongly structured documents and aim to investigate techniques for finding information using queries considering content and structure. In the following we will present the two main collections with their respective evaluation measures. Finally we will present our results over the two associated tracks.

3.1 Collections and Evaluation Metrics

INEX 2005 SSCAS Track. The INEX 2005 collection is composed of about 12000 XML documents from the IEEE Computer Society scientific papers. These documents have an average of 1500 elements for a hierarchical depth of 6.9. In total there are 8 millions nodes and 192 different tags. An example of an IEEE document can be found in figure 4.

Two main types of queries are available, namely *Content Only* (CO) and *Content And Structure* (CAS). Tasks using CAS queries are centered on structural constraints. Four subtasks were proposed. To evaluate queries in which structural constraints are semantically relevant, we use the SSCAS subtask, in which 8 queries specify strict constraints on the target element (the element we want to retrieve) and its environment. The adequacy between document and query hierarchical structure). This SSCAS task was not reconducted in the following INEX campaigns, until 2010 when the Datacentric track began. The CAS subtasks use two metrics [13]: *Non-interpolated mean average effort-precision* (MAeP) which

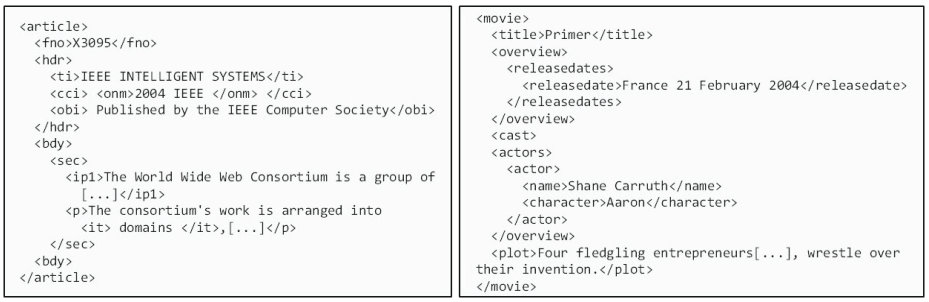

Fig. 4. Examples of documents from the IEEE (left) and IMDB (right) collections

is used to average the effort-precision measure at each rank and *Normalized cumulated gain* (nxCG).

INEX 2010 Datacentric Track. This track usesthe IMDB data collection generated from IMDB web site. In total, the data collection contains 4,418,081 XML files, including 1,594,513 movies, 1,872,471 actors, 129,137 directors who did not act in any movie, 178,117 producers who did not direct nor act in any movie, and 643,843 other people involved in movies who did not produce nor direct nor act in any movie. An example of an IMDB document can be seen in figure 4. 28 queries are associated with the collection. As IMDB uses two DTDs we created three graphs : one for movie, one for person and one merged on nodes shared by the previous two. The merged DTD will be used for queries where relevant nodes could be found either in a movie or in a person.

Effectiveness of SIR systems was evaluated though two measures [27] detailed in [12] : *MAiP* (Mean average interpolated precision) is computed through 101 standard recall points (eg : 0.00, 0.01, etc..) and *MAgp T2I* assesses the exhaustivity of the returned results. Element score is the score at a tolerance to irrelevance (T2I) points, 300 in our case with no overlap.

3.2 Results

Our first aim is to evaluate the effectiveness of TED with DTD based costs. To do so we set two different costs versions. The first one is *TED with DTD* in which the costs of removing and relabeling are evaluated from the DTDs of the different collections (as explained in section 2.2). The second one is *TED without DTD* in which the costs are set to 0 for a relabeling of two nodes sharing the same label and 1 otherwise. Removing is set to 0.5 for nodes whose label is in the query and 1 otherwise. These latter scores are inspired from previous work [15]. The rest of this section is structured as follows: first we will present our results on each collection for various settings of the equation [7] λ parameter. We will then present our results based on INEX benchmarks compared to the official INEX participants best results on these tracks.

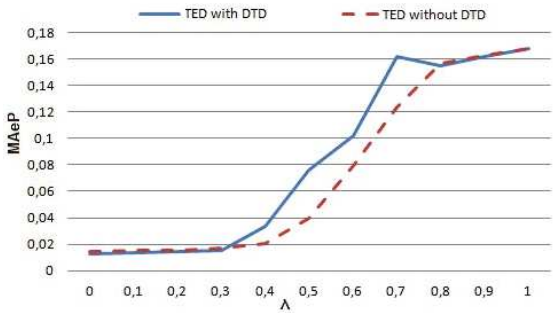

Fig. 5. Results for the MAeP measure with various values of λ, INEX 2005

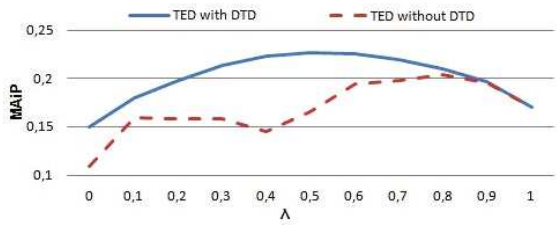

Fig. 6. Results for the MAiP measure with various values of λ, INEX 2010

TED for Structure Evaluation. Our INEX 2005 SSCAS task results for the whole λ spectrum on MAeP measure are presented in figure 5. It appears that TED always reduces effectiveness (results are better with $\lambda = 1$). We however notice that performances are better with the DTD based costs for λ between 0.3 and 0.7. Indeed if structure helps during the score propagation process from the leaves to inner nodes, it could however harms the search process when structure does not return any relevant elements by itself. The low results obtained under $\lambda = 0.3$ show that the structure does not provide an answer to the query without a proper content evaluation and does not provide semantic information.

Regarding INEX 2010 Datacentric task, our results for various values of the λ parameter over the MAiP measure are shown in figure 6. Contrary to INEX 2005, it appears that combining content and structure improves significantly the results for both runs. Moreover, our DTD based costs run scores significantly higher than the one with the costs set empirically. Another observation is that contrary to the IEEE collection a strongly structured based combination tends to return results even for the lowest values of λ.

At first sight our results may appear contradictory on the overall structure usefulness. However, as there has been an extensive controversy in the literature about similar issues [25] we will discuss in the next section what we believe could be the reason of such results.

Table 1. Results for $\lambda = 0.7$ compared with official participants best results for the INEX SSCAS track, strict quantization

Runs	**MAeP**	**nxCG10**	**nxCG25**	**nxCG50**
TED with DTD	**0.1622**	0.425	0.4	0.36
MaxPlanck	0.1334	**0.45**	0.3956	0.3894
TED without DTD	0.1235	0.425	0.38	0.365
IBMHaifa	0.1023	0.225	**0.4278**	**0.4067**

Table 2. Results for $\lambda = 0.7$ compared with official participants best results, INEX 2010

Runs	**MAiP**	**Runs**	**MAgP**
TED with DTD	**0.2197**	OTAGO-DC-BM25	**0.2491**
TED without DTD	0.1984	UPFL15TM	0.2458
ufam2010Run2	0.1965	UPFL15TMImov	0.2433
UPFL15TMI	0.1809	Kasetsart	0.1811
UPFL15TMImo	0.1752	TED with DTD	0.1335
ufam2010Run1	0.1614	TED without DTD	0.1183

Overall Performances. In order to compare our approach with INEX official participants we fix $\lambda = 0.7$ in equation (7) as it is the best compromise in term of results for our two runs.

Table 1 shows our results on INEX 2005 compared to the best participants (the Max Planck institute with its TopX [24] system which uses a database approach and IBM Haifa Research Lab). Our method outperforms state-of-the-art methods on the MAeP metric and obtains similar results on the nxCG metric.

Table 2 presents our results on INEX 2010 for both official metrics. Among the official participants, one can cite the Otago university which used a BM25 trained on INEX 2009 [10] and a divergence language model [2] (run OTAGO-DC-BM25) and Pompeu Fabra [21] which used a language model (runs UPFL*).

Our approaches score better than the official participants when using the TED part on the *MAiP* measure (+12 % on MAiP for the DTD based costs and +1% for our empirically set costs compared to the first ranked run in the official results (Ufam)). However our *MAgP* score is significantly lower (we should have been ranked 6^{th}). This can be explained by the fact that the *MAgP* over-ranks systems returning whole documents instead of elements.

4 Structure Usefulness and Collection Type

As we have seen in section 1, the insight behind SIR models is that document structure owns information that could help to improve the search process. However in literature there is a controversy on the real usefulness of structure itself. While some authors such as Sauvagnat et al. [22] outline its importance, others like Trotman et al. [26] or recent INEX track overview [27] and [28] show its very low usefulness. Overall Trotman et al. [25] stated that structural constraints

Table 3. Structural constraints distribution over three INEX ad-hoc tracks

Tracks	1 constraint	2 constraints	$\geq$ 3 constraints
INEX 2005 SSCAS	**25%**	37.5%	37.5%
INEX 2010 Datacentric	12%	**48%**	**40%**

helps in half of the cases. We believe that this controversy and our contradictory results could be explained by the nature of the collections themselves.

In the IEEE 2005 collection, the nested tags are not informative as they only represent a structure similar to what could be found in a book. We also noticed that the leaves text content contains terms similar to ones that can be found in a narrative description. Thus structure is more a filter and a support for the final ranking that a mandatory part in the search process. In the IMDB collection, structure carries semantic and is as effective as the text content to answer the query. For example in figure 1 if a user wants to retrieve the title of a movie directed by Terry Gilliam he will only be interested by Terry Gilliam director and not Terry Gilliam cameo in a movie (under the element labeled "director" and not "trivia"). A second observation is that the text in leaves is short and focus (with the exception of some elements like "plot" most of the leaves only contain three or four terms). Thus the structure is as important as the content itself.

Structured documents can be classified according to two main categories. A document can be *text-oriented*, meaning that the text is exhaustive and the tags structural organization could be used as a filter; or it can be *data-oriented* with very specific text content split over semantically meaningful tags. The collection we used in these experiments are *text-oriented* for IEEE and *data-oriented* for IMDB. Our results could then be explained with the previous category the collection falls into : structure should be more strictly followed on a data-centered collection than on a text-centered one.

If the collection type can explain our results and the overall controversy we cannot, however, discard query formulation. It appear that CAS queries structural constraints are mostly loosely set. Trotman et al. [26] explain the lack of improvement over the structure by the inability of the users to give relevant structural hints. Another lead could be the short amount of structural constraints. Following table 3 it is surprising to realize that 62.5% of the query in INEX 2005 SSCAS task contain less than 3 structural constraints with 25% of them containing only 1.

5 Conclusions and Future Work

In this paper we presented a SIR model based on Tree-Edit Distance (TED) to measure the structural similarity between a query and a document subtree. As TED cost impacts the effectiveness of the search process we proposed to setup these costs according to the DTD. The results showed that TED performances depend on the type of collections, i.e. *text-oriented* or *data-oriented*. Indeed TED is more useful for a data-oriented collection than for a text-oriented, where structure is only considered as a hint.

In future work we plan to further study terms and tags distribution on some other XML collections. We also plan to improve the content scoring part of our model as it acts as a baseline to measure the efficiency of our structure approach.

References

1. Alilaouar, A., Sedes, F.: Fuzzy querying of XML documents. In: Web Intelligence and Intelligent Agent Technology Conference, France, pp. 11–14 (2005)
2. Amati, G., Van Rijsbergen, C.J.: Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans. Inf. Syst. 20, 357–389 (2002)
3. Barros, E.G., Moro, M.M., Alberto, H., Laender, F.: An Evaluation Study of Search Algorithms for XML Streams. JIDM 1(3), 487–502 (2010)
4. Ben Aouicha, M., Tmar, M., Boughanem, M.: Flexible document-query matching based on a probabilistic content and structure score combination. In: Symposium on Applied Computing (SAC), Sierre, Switzerland (March 2010)
5. Bender, M.A., Farach-Colton, M.: The LCA Problem Revisited. In: Gonnet, G., Panario, D., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 88–94. Springer, Heidelberg (2000)
6. Bille, P.: A survey on tree edit distance and related problems. Theoritical Computer Science 337(1-3), 217–239 (2005)
7. Damiani, E., Oliboni, B., Tanca, L.: Fuzzy techniques for XML data smushing. In: Proceedings of the International Conference, 7th Fuzzy Days on Computational Intelligence, Theory and Applications, pp. 637–652 (2001)
8. Dulucq, S., Touzet, H.: Analysis of Tree Edit Distance Algorithms. In: Baeza-Yates, R., Chávez, E., Crochemore, M. (eds.) CPM 2003. LNCS, vol. 2676, pp. 83–95. Springer, Heidelberg (2003)
9. Floyd, R.W.: Algorithm 97: Shortest path. Commun. ACM 5, 345 (1962)
10. Jia, X.-F., Alexander, D., Wood, V., Trotman, A.: University of Otago at INEX 2010. In: Geva, S., Kamps, J., Schenkel, R., Trotman, A. (eds.) INEX 2010. LNCS, vol. 6932, pp. 250–268. Springer, Heidelberg (2011)
11. Sparck Jones, K.: Index term weighting. Information Storage and Retrieval 9(11), 619–633 (1973)
12. Kamps, J., Pehcevski, J., Kazai, G., Lalmas, M., Robertson, S.: INEX 2007 Evaluation Measures. In: Fuhr, N., Kamps, J., Lalmas, M., Trotman, A. (eds.) INEX 2007. LNCS, vol. 4862, pp. 24–33. Springer, Heidelberg (2008)
13. Kazai, G., Lalmas, M.: INEX 2005 Evaluation Measures. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 16–29. Springer, Heidelberg (2006)
14. Klein, P.N.: Computing the Edit-Distance between Unrooted Ordered Trees. In: Bilardi, G., Pietracaprina, A., Italiano, G.F., Pucci, G. (eds.) ESA 1998. LNCS, vol. 1461, pp. 91–102. Springer, Heidelberg (1998)
15. Laitang, C., Pinel-Sauvagnat, K., Boughanem, M.: Edit Distance for XML Information Retrieval: Some Experiments on the Datacentric Track of INEX 2011. In: Geva, S., Kamps, J., Schenkel, R. (eds.) INEX 2011. LNCS, vol. 7424, pp. 138–145. Springer, Heidelberg (2012)
16. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady 10, 707 (1966)

17. Mehdad, Y.: Automatic cost estimation for tree edit distance using particle swarm optimization. In: Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, ACLShort 2009, pp. 289–292 (2009)
18. Neuhaus, M., Bunke, H.: Automatic learning of cost functions for graph edit distance. Information Science 177(1), 239–247 (2007)
19. Oncina, J., Sebban, M.: Learning stochastic edit distance: Application in handwritten character recognition. Pattern Recogn. 39, 1575–1587 (2006)
20. Popovici, E., Ménier, G., Marteau, P.-F.: SIRIUS: A Lightweight XML Indexing and Approximate Search System at INEX 2005. In: Fuhr, N., Lalmas, M., Malik, S., Kazai, G. (eds.) INEX 2005. LNCS, vol. 3977, pp. 321–335. Springer, Heidelberg (2006)
21. Ramírez, G.: UPF at INEX 2010: Towards Query-Type Based Focused Retrieval. In: Geva, S., Kamps, J., Schenkel, R., Trotman, A. (eds.) INEX 2010. LNCS, vol. 6932, pp. 206–218. Springer, Heidelberg (2011)
22. Sauvagnat, K., Boughanem, M., Chrisment, C.: Why Using Structural Hints in XML Retrieval? In: Larsen, H.L., Pasi, G., Ortiz-Arroyo, D., Andreasen, T., Christiansen, H. (eds.) FQAS 2006. LNCS (LNAI), vol. 4027, pp. 197–209. Springer, Heidelberg (2006)
23. Tai, K.-C.: The tree-to-tree correction problem. J. ACM 26, 422–433 (1979)
24. Theobald, M., Schenkel, R., Weikum, G.: Topx XXL. In: Proceedings of the Initiative for the Evaluation of XML Retrieval, pp. 201–214 (2005)
25. Trotman, A.: Processing structural constraints. In: Encyclopedia of Database Systems, pp. 2191–2195 (2009)
26. Trotman, A., Lalmas, M.: Why structural hints in queries do not help XML-retrieval. In: SIGIR 2006, pp. 711–712 (2006)
27. Trotman, A., Wang, Q.: Overview of the INEX 2010 Data Centric Track. In: Geva, S., Kamps, J., Schenkel, R., Trotman, A. (eds.) INEX 2010. LNCS, vol. 6932, pp. 171–181. Springer, Heidelberg (2011)
28. Wang, Q., Ramírez, G., Marx, M., Theobald, M., Kamps, J.: Overview of the INEX 2011 Data-Centric Track. In: Geva, S., Kamps, J., Schenkel, R. (eds.) INEX 2011. LNCS, vol. 7424, pp. 118–137. Springer, Heidelberg (2012)